Phase 4 Report

2.1 The Game

- Brief overall description of the game

Game Overview:

The goal of the Main Character (Student) is to graduate from SFU. In other words, the Main Character wants to escape from SFU, the setting of the game. Our Main Character will start in a lecture room. The Protagonist must gain knowledge during their time in SFU by reading books and reviewing notes. In order to win the game, which is to escape SFU, the player has to collect all of the Static Rewards (Textbooks) and Bonus Rewards (Cheat Sheets) in order to gain a high enough Score (Course Credits). However, the Main Character (Student) needs to avoid the Moving Enemy (Professor/Raccoons) and Static Enemy (CCTV), which implies that whether or not they graduate will depends on what grade the Professor will give in order to pass the course, and whether they could stay healthy throughout the SFU years without getting attacked by the Raccoons. On another hand, CCTV will symbolize whether the Main Character (Student) will break any school rules or not obey the pledge of academic dishonesty. The goal of the Main Character (Student) is to not get caught by Enemies and collect enough Rewards before reaching the End Goal. The game world is based on SFU, thus, the game will start in the lecture room inside the AQ, once completed, the Main Character (Student) will proceed to outside of AQ, then to the Bus Loop to end the Game.
PS. We want to make the game as realistic, original and relatable as possible.

- Discuss how much you have been faithful to your original plan and design

In Phase 1, we had a couple of ideas, such as:

The first game, Temple Run, features a treasure hunter as the main character who must navigate through an ancient temple while avoiding enemies such as bats, spiders, and thieves. There are also hidden traps and mines that serve as punishments. The player can collect gold and rare artifacts as normal rewards, as well as bonuses such as buffs, speed increases, and defense boosts. The game is set in an ancient temple and the objective is to steal as much gold as possible while avoiding enemies and escaping without getting killed.

The second game, Prison (SFU) Break, has an inmate as the main character who must escape from a prison while evading prison guards who act as moving enemies. There are also security cameras that serve as static enemies. The player can collect keys, tools, and money as normal rewards, as well as bonuses such as speed increases. The prison setting includes cells, corridors, and outdoor areas, and the objective is to escape the prison without being caught by the guards or cameras.

Nevertheless, we decided on SFU Break as the concept excited us. As for the design of the game, we initially planned on building the game using the:

MVC Pattern that separates the Game Logic into 3 Components:

1. The Model: Game State and Logic

2. The View: Displays the Game State to the User
3. The Controller: Handles User Input and Updates (Model) and (View)

- The Model:
    - Game World (Board of the Game)
    - Main Character, Enemies, Rewards
- The View:
    - The Graphics and User Interface
- The Controller:
    - Handles Keyboard Input

However, it seemed too complicated to implement considering our lack of experience with Java. For example, the UML Diagram that we designed in Phase 1 highlighted our initial shortcomings with Java and our structural designing-skills required for the game. We thought we could build the whole game with just 5 main classes, but after we completed the game, we knew it was poorly thought-out, and acknowledged that our initial designs were inadequate. In short, we did not stay faithful to our initial plan and we made sizable changes and improved the design in every phase of the project.

In the end, we implemented the Abstract Factory method as it is easier to implement since we have learned it in class before.

· How the final product varies from that plan and justify what has changed

As for the final product, we managed to keep the majority of the plan, especially in the narrative of the game. Having said that, we made large changes in the structure and technology aspects of the game throughout the process.

- We used the Abstract Factory method instead of the MVC method.
- The design of our game objects and characters improved drastically.
- The code of our game has been refactored multiple times as we are improving in our Java proficiency.
- The UML diagram is totally different from what we had initially, since we have almost 20-30 classes.
- The UI also had some modifications as we decided not to include the difficulty level in our game to make it more like a storytelling game.

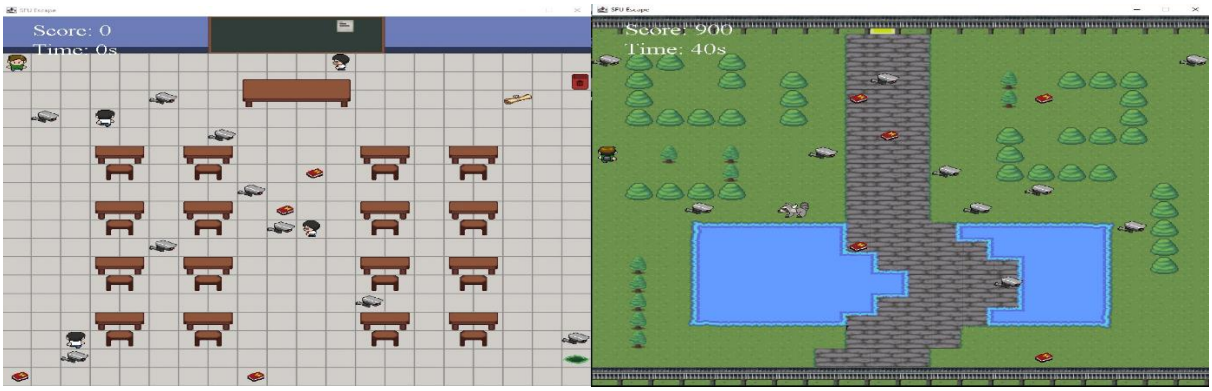
· What are the most important lessons you have learned

As far as project and team management is concerned, the two main areas of improvement that we highlighted were communication and experience. We initially believed we had put in an adequate amount of effort to ensure each team member was aware of their relevant goals and objectives. However, we have since learned that we could have emphasized communication even more. We decided early on to adopt an agile software development approach. To improve our communication, we should have appreciated the need to constantly discuss and review each of our perspectives on development. We have learned that this is crucial for managing the chaos inherent in the software development approach. Secondly, we have learned of the importance of

experience with every aspect of developing a program. Especially when it came to implementation, we spent a lot more time developing and testing code than we would have if we initially had a higher comprehension of Java.

The main aspect of the software development process that we value more now is the initial design and planning phase. We found that the later we tried to change elements of the program, the more time-consuming it became; as the game was further developed, altering the design meant having to consider and revise an increasing number of code blocks. If we spent more time thinking about and researching for the initial design and planning phase, we probably would have detected and considered a lot of the problems that we ended up facing. This was especially significant when it came to integrating different aspects of the program. When reflecting, it was very apparent that we underestimated the cost of integration. Another aspect of the software development process that we did learn to value more was testing. Planning for testing, especially considering cohesion, was something we should have done more. It became difficult to distinguish the exact issues that were shown in our testing phase. This led to us spending a good amount of time adjusting our code to make it more testable.

2.2 Tutorial



As seen in the screenshot above, the game begins with the user (left picture, top left in green) in a classroom with desks, professors, books, security cameras, and a red trash square on the right side. Books are +100 to the user's score while cheat sheets (which appear sporadically) are +200. Security cameras apply a negative to the user's score, which must be kept positive for the game to continue. Although there are 2 maps for this game, both follow the same goal of collecting all the books in the current map without getting a negative score or coming in contact with any of the moving enemies. The main character is controlled by the arrow keys on the keyboard and are the only controls needed for the game. A scenario that is important to note is when the main character comes in contact with other objects in the game such as books, enemies (static/moving), desks, and walls. In the event that the main character touches a moving enemy, the user has lost and the game is over. This is in contrast to if the main character touches a static enemy - the game could continue, but the user will have a negative applied to their score. If the score falls below 0 as a result of the encounter, the game is lost as a negative score automatically ends the game. The user is not able to go through objects such as tables or trees and the user is stonewalled if they attempt to go through any of these types of objects. Another important scenario is when the user is able to go through to the next level. This can be done in level 1 when the user has amassed more than 500 points. The user must then go through the green

portal at the bottom right of level 1 to be put through to level 2. If the user has not gotten more than 500 points yet, they will not be able to go through to level 2 and so nothing will happen if they go to the portal with less than 500 points. Similarly in level 2, the user must get more than 1000 points before leaving the game through the yellow exit seen at the top of level 2's map (picture on right).

A demo of the game can be viewed at the following link:

https://www.youtube.com/watch?v=_xqhko1xM-s&ab_channel=%EC%A1%B0%EC%84%B1%ED%99%98

In summary, our group had many different ideas when we began this project, as well as many ideas on how to implement our vision. We soon found that adjustments had to be made to our initial plan once we realized the nature of our game and so we drastically changed our implementation to accommodate our plans. Although many structural changes were made throughout our development timeline, our vision for the game held up through all phases of this project. We learned lessons in communication and the importance of the initial planning phase, as well as testing, for a project of this size which we will carry forward in our future when developing software of all types.